

# Optimal Reduction without Oracle?

Anton Salikhmetov

January 24, 2017

## Abstract

We suggest an approach to solve the problem of matching fans in interaction net implementations of optimal reduction for the pure untyped lambda calculus without use of any additional agent types. Our implementation supports a wider class of lambda terms than the abstract version of Lamping’s algorithm and beats the interaction net implementation of closed reduction by the total number of interactions.

## 1 Problem

Matching fans is the main problem of implementation of optimal reduction in interaction nets [1]. Existing solutions use so-called oracle which is implemented using bracket and croissant agents in BOHM [1] and delimiter agents in Lambdascope [2]. Unfortunately, both versions produce significant overhead.

Since that overhead is due to the oracle, we decided to limit our signature to the basic types only (abstraction, application, erase, and fan) and experiment with more sophisticated structures attached to fans rather than just integer indices, aiming at the same behavior of agents as if the oracle were still present.

Our experiments resulted in a partial solution we present in this paper. It benefits from optimal number of  $\beta$ -reductions like BOHM and Lambdascope. Simultaneously, it has performed better by the number of interactions than non-optimal CVR [3].

## 2 Solution

We work in interaction calculus [4]. The signature of our interaction system is

$$\Sigma = \{ @, \lambda, \varepsilon \} \cup \{ \delta_t^n \mid t \in \mathbb{T}, n \in \mathbb{N} \},$$

elements of  $\mathbb{T}$  being inductively defined as two-colored binary trees with names in each leaf:

$$t ::= x \mid \langle t_1 + t_2 \rangle \mid \langle t_1 - t_2 \rangle.$$

Intuitively, one can think of  $n$  in  $\delta_t^n$  as number of this agent’s instances created during its life, and  $t$  can be thought of as this agent’s identity.

The interaction rules are as follows:

$$\begin{aligned}
\varepsilon &\bowtie \alpha[\varepsilon, \dots, \varepsilon] \quad (\forall \alpha \in \Sigma); \\
@[x, y] &\bowtie \lambda[x, y]; \\
\delta_t^n[\lambda(x, y), \lambda(v, w)] &\bowtie \lambda[\delta_t^{n+1}(x, v), \delta_t^{n+1}(y, w)]; \\
\delta_t^n[@(x, y), @(v, w)] &\bowtie @[\delta_t^{n+1}(x, v), \delta_t^{n+1}(y, w)]; \\
\delta_t^m[x, y] &\bowtie \delta_t^n[x, y]; \\
\delta_t^n[x, y] &\bowtie \delta_u^n[x, y]; \\
\delta_t^m[\delta_{\langle u+t \rangle}^n(x, y), \delta_{\langle u-t \rangle}^n(v, w)] &\bowtie \delta_u^n[\delta_t^{m+1}(x, v), \delta_t^{m+1}(y, w)] \quad (t \neq u, m > n).
\end{aligned}$$

The last rule in the list above is central for this paper. Our intuition for this rule is that the fan with greater number of instances duplicates another fan with different identity. Here,  $\langle u \pm t \rangle$  can be thought of as the left/right clone of  $u$  created by  $t$ .

Translation of  $\lambda$ -terms to interaction nets is done the same way as in other optimal implementations, except that each bracket, croissant, or delimiter is replaced with just a wire, and each fan is represented as a  $\delta_x^1$  agent with a unique name  $x$ .

In order to implement interaction rules in  $O(1)$  by time and space, trees can be replaced with hashes, if we represent names as strings. Specifically, we suggest the following mapping  $S(t)$  from trees to hashes, where  $\circ$  is string concatenation and  $H$  is SHA-256 or any other hash function that is resistant to collisions:

$$\begin{aligned}
S(x) &= H('x'); \\
S(\langle t_1 + t_2 \rangle) &= H(S(t_1) \circ '+' \circ S(t_2)); \\
S(\langle t_1 - t_2 \rangle) &= H(S(t_1) \circ '-' \circ S(t_2)).
\end{aligned}$$

Then, both  $\langle t_1 \pm t_2 \rangle$  operations and equality test for trees become  $O(1)$  in time and space.

### 3 Prototype

The following table shows some evaluations of Church numerals and includes benchmarks for the larger examples from [3], providing comparison between CVR and a prototype that implements the solution suggested in this paper. As usual for such kind of benchmarks, results of the form  $N(N_\beta)$  should be read as total of  $N$  interactions, of which  $N_\beta$  were  $\beta$ -reductions in the sense of  $@ \bowtie \lambda$  interactions.

Term	CVR	Prototype
2 2 2 10 <i>I I</i>	1058(179)	707(67)
3 2 2 2 <i>I I</i>	3992(542)	1158(40)
10 2 2 <i>I I</i>	15526(2082)	4282(56)
4 2 2 2 <i>I I</i>	983330(131121)	262377(61)
2 2 2 2 10 <i>I I</i>	4129050(655410)	2359780(198)

The prototype is being developed in the context of our ongoing project currently focused on obtaining an efficient token-passing optimal reduction with embedded read-back [5]. This allowed us to use the prototype on essentially  $\lambda K$ -terms and compare them against token-passing version of Lambdascope with embedded read-back. As our main test case, we used a complex  $\lambda K$ -term representing arithmetical expression  $3^3 - (2+2)!$  with Church numerals and factorial defined via Turing’s fixed point combinator. With the same waiting construct and read-back mechanisms in use, the benchmarks are as follows: 16024(778) for the prototype and 813753(778) for Lambdascope.

## 4 Further work

To the best of our knowledge, the suggested mechanism to track identities of fans is a new approach. This mechanism partially eliminates the need in the oracle. However, the naive level-tracking part (meant to decide which one of two interacting fans with different identities is active) suffers from counterexamples. Perhaps, the simplest one is  $\omega (\lambda x. \omega (\lambda y. y x))$ , where  $\omega = \lambda x. x x$ . Evaluating that term results in a fan reaching the interface of the net.

Still, we find the suggested approach promising, and believe that it could be possible to improve the level-tracking mechanism in order to block counterexamples and support all  $\lambda K$ -terms, thus achieving the universal version of Lamping’s abstract algorithm.

## References

- [1] A. Asperti and S. Guerrini.  
*The optimal implementation of functional programming languages.*  
 Cambridge Tracts in Theoretical Computer Science No. 45.  
 Cambridge University Press, 1998.
- [2] V. van Oostrom, K.-J. van de Looij, and M. Zwitterlood.  
 “Lambdascope: another optimal implementation of the lambda-calculus”.  
 In: *Workshop on Algebra and Logic on Programming Systems (ALPS)*. 2004.
- [3] I. Mackie. “An interaction net implementation of closed reduction”.  
 In: *Symposium on Implementation and Application of Functional Languages*.  
 Springer. 2008, pp. 43–59.
- [4] M. Fernández and I. Mackie. “A calculus for interaction nets”.  
 In: *Principles and Practice of Declarative Programming*. Springer, 1999, pp. 170–187.  
 DOI: 10.1007/10704567.
- [5] A. Salikhmetov. “Token-passing Optimal Reduction with Embedded Read-back”.  
 In: *Proceedings 9th International Workshop on Computing with Terms and Graphs*,  
 Eindhoven, The Netherlands, April 8, 2016. Vol. 225.  
 Electronic Proceedings in Theoretical Computer Science.  
 Open Publishing Association, 2016, pp. 45–54. DOI: 10.4204/EPTCS.225.7.